CMS//
2006-07-14
D R A F T

Proposal for Web Based Monitoring and Database Browsing

Francesca Cavallari,
INFN, Rome

William Badgett, Steve Murray
Fermilab

Ricky Egeland
University of Minnesota

Frank Glege
CERN

Zongru Wan
Kansas State University

Abstract

The CMS Online Database contains a wealth of important information regarding the current and past states of the CMS detector. These data are not readily viewable from outside the P5 CMS experimental site due to security restrictions. In this document we propose a light-weight web based interface to the CMS detector status information in the database designed for users viewing data remotely. The proposed protocol is simple HTTP with Tomcat/Jakarta hooks on the server, but with no special options required on the web client browser side. The output format of the data would consist of histograms, scatter plots, and web tables viewable on the client's web browser; and text, Root and XML based files downloadable via links produced in the web interface. To facilitate the browsing functionality, we propose a set of meta-data tables to be established describing the various monitoring tables in the database.

1. Introduction

2. Proposal for meta-data for the PVSS/DCS custom table

The PVSS system provides a standardized set of Oracle database tables to hold channel sensor definitions and history of changes in those channels. As such, we have provided a general purpose web based browser that identifies the existence of such tables in the CMS omds database and presents them to the user in the form of a tree:

http://cmsdaq.cern.ch/cmsmon/cmsdb/servlet/SlowControlBrowser

The top level branch of the tree is the database schema owner; subsequent lower level leafs are a tokenized version of the name of the sensors based on the contents of the PVSS ELEMENTS table.

However, some sub-detectors (e.g. ECAL) may opt out of using the prepackaged PVSS tables in favor of a more convenient detector specific set of tables. There is no way *a priori* method for a database display application to know the existence and contents of such tables. Therefore, we propose to implement a set of CMS standard *meta-data* database tables that describe the existence and contents of the sub-detector specific tables.

2.1 CONDITIONS_TABLE_DESCRIPTOR Table

The following entry describes the "CONDITIONS_TABLE_DESCRIPTOR Table" which establishes what tables have the "Value vs. Time" structure. The pair of columns (OWNER, TABLE_NAME) must be unique within this table. The primary key ID forms a foreign key pointing to the ID_TABLE column of the COLUMN_DESCRIPTOR table described later. The TIME_COLUMN entry tells which column of the described table contains a time stamp for the row; Oracle allows three main time stamp types, and the TIME_DATATYPE column must specify which one, or whether a simple NUMBER is used. Likewise, RUNNUMBER_COLUMN indicates (optionally) which column in the described table contains the run number as created in the CMS_RUNINFO *omds* database schema, and this number must be an integer.

Some tables will have columns which correspond to multiple physical sources of data; to distinguish between these possibilities, use the POSITION_COLUMN to point to the appropriate column.

TABLE_DESCRIPTOR Table		
Column	**Type**	**Description**
ID	NUMBER(22) primary key	Primary key for table, arbitrary integer
ID_SOURCE	NUMBER(22) foreign key optional	If this table entry is grouped by the source of data, then this column contains a pointer to a CONDITIONS_SOURCE_ DESCRIPTOR table row.
OWNER	VARCHAR2 not null; unique	Schema owner of table being described
TABLE_NAME	VARCHAR2 not null; unique with OWNER	Table name where the data area held.
SYSTEM	VARCHAR2 not null	Specifies which sub-detector or sub-system is the source of data
DESCRIPTION	VARCHAR2 optional	Human readable text description of sensor
LABEL	VARCHAR2 optional	Short but human readable label for table display purposes
IS_ENABLED	NUMBER(1) not null	Flag to indicate current enable status of the table
IS_PRIVATE	NUMBER(1) Not null	Flag to indicate this table is not for the general CMS public
TIME_COLUMN	VARCHAR2	Name of the table column that contains the time stamp of the row data
RUNNUMBER_COLUMN	VARCHAR2 optional	Name of table column that contains the CMS_RUNINFO run number value

2.2 CONDITIONS_SOURCE_DESCRIPTOR Table

As an additional layer of hierarchy, sub-detectors may wish to group their tables according to the source of the data. The following table describes the CONDITIONS_SOURCE_DESCRIPTOR table. Each row in this table describes a group of tables in the CONDITIONS_TABLE_DESCRIPTOR

table. The primary key ID of the CONDITIONS_SOURCE_DESCRIPTOR table points to the ID_SOURCE column of the CONDITIONS_TABLE_DESCRIPTOR table.

SOURCE_DESCRIPTOR Table		
Column	**Type**	**Description**
ID	NUMBER(22) primary key	Primary key for table, arbitrary integer
SOURCE_NAME	VARCHAR2 not null; unique	Grouping of tables within a sub-detector, according to the source which generated the data
DESCRIPTION	VARCHAR2 optional	Human readable description of this source of data
IS_ENABLED	NUMBER(1) not null	Flag to indicate current enable status of the source of data
IS_PRIVATE	NUMBER(1) Not null	Flag to indicate this source is not for the general CMS public

2.3 CONDITIONS_COLUMN_DESCRIPTOR Table

The following entry defines the CONDTITIONS_COLUMN_ DESCRIPTOR, whose purpose is to specify which columns of the target table are interesting for viewing purposes. In this table, the (ID_TABLE, COLUMN_NAME) pair must be unique, and the ID_TABLE column must point to an entry in the TABLE_DESCRIPTOR_TABLE described above. A foreign key constraint will be implemented to enforce this rule.

COLUMN_DESCRIPTOR Table		
Column	**Type**	**Description**
ID	NUMBER(22) primary key	Primary key of this table; arbitrary integer
ID_TABLE	NUMBER(22) Foreign key	Primary key of the parent TABLE_ DESCRIPTOR table

OWNER	VARCHAR2 not null	Owner of target data table
TABLE_NAME	VARCHAR2 not null	Table name of target data table
COLUMN_NAME	VARCHAR2 not null; unique	Column name within the TABLE_NAME where the data are kept; together with OWNER, TABLE_NAME and COLUMN_NAME, must be unique
UNITS	VARCHAR2 optional	Units in which the sensor values are measured; could be constrained to set of known values (could be separate table)
HARDWARE_ID	NUMBER Optional	CMS Hardware ID, if applicable
DESCRIPTION	VARCHAR2 optional	Human readable text description of sensor
LABEL	VARCHAR2 optional	Short human readable text for display purposes
TARGET_DATA_TYPE	VARCHAR2 optional	Data type to be used when converting from database value to program value
BEGIN_VALID	DATE optional	Absolute time after which data are valid
END_VALID	DATE optional	Absolute time before which data are valid
POSITION_A_COLUMN	VARCHAR2 optional	Defines column which holds one position variable to fully specify data
POSITION_A_MIN	NUMBER	Minimum value of A
POSITION_A_MAX	NUMBER	Maximum value of A
POSITION_B_COLUMN	VARCHAR2	Defines another column which holds one position variable to fully specify data
POSITION_B_MIN	NUMBER	Minimum value of B
POSITION_B_MAX	NUMBER	Maximum value of B
IS_ENABLED	NUMBER(1) not null	Flag to indicate current enable status of the sensor, value 0 or 1 (default)

IS_PRIVATE	NUMBER(1) not null	Flag to indicate whether these data are for public CMS display value 0 (default) or 1
IS_PLOTTABLE	NUMBER(1) not null	Flag to indicate whether data may be easily plotted for display, value 0 or 1 (default)
MIN_WARNING_THRESHOLD	REAL, optional	Warning min threshold
MAX_WARNING_THRESHOLD	REAL, optional	Warning max threshold
MIN_ERROR_THRESHOLD	REAL, optional	Error min threshold
MAX_ERROR_THRESHOLD	REAL, optional	Error max threshold

2.4 CONDITIONS_LAST_VALUE Table

In terms of current time monitoring, it is extremely useful to have an easily accessible record of what the most recent value of a measurement was. To implement this possibility, we describe the CONDITIONS_LAST_VALUE table below. The primary key is formed by the pair of values (ID, POSITION). The ID column points back to the primary key of the CONDITIONS_COLUMN_DESCRIPTOR table. If there are multiple physical sources of data, then the POSITION column can be used to distinguish between them. If there is just one source of data, the POSITION column can be set to zero.

Filling of this table is envisioned to be mostly via automatic methods. When a row in the CONDITIONS_COLUMN_DESCRIPTOR table is inserted, a database trigger will automatically insert a corresponding row(s) in the CONDITIONS_LAST_VALUE table. Correspondingly, a trigger may be created on the target data table to update the CONDITIONS_LAST_ VALUE table when an insert takes place.

LAST_VALUE Table		
ID	NUMBER; With POSITION, forms primary key	Integer pointing to entry in Column Descriptor Table

POSITION_A	NUMBER Not null; with ID forms primary key	If the column represents multiple physical sensors, use this column to denote them; set to zero if not used.
POSITION_B	NUMBER Not null; with ID forms primary key	Secondary column to denote multiple physical sensors; defaults to 0
LAST_VALUE	REAL, optional	Last value read from sensor or other source; could be filled by INSERT trigger on data table
LAST_TIME	TIMESTAMP, optional	Time the LAST_VALUE was taken
LAST_STATUS	NUMBER(11) optional	Status of last measurement per PVSS, or particular subsystem if not PVSS

2.5 Creating and Filling Meta-data Tables

The set of meta-data tables could be unique within the database, or at least unique within a given database schema owner. With the case of multiple sets of meta-data tables, the names of the tables should be the same across schemas, with the schema owner identifying the subsystem. We propose implementing these four tables in all schemas in the *omds* database. This method automatically gives the database schema owner control over the contents as well as the ability to extend the meta-data tables.

The *ID* integer-type column has been implemented as a simple primary key in order to make joins between the tables faster for the database SELECT table-join statements. Also, the standard replication methods prefer the simple primary key. But as such, the ID value has little information content for the user of the table, and the user will not be responsible for creating the value and should not care what the value turns out to be. A global database sequence will be used to provide unique ID values across the database,

accessible by all the schema owners and generated by a database trigger at INSERT time.

In practice, the ID_TABLE number must be known at INSERT time when writing to the CONDITIONS_COLUMN_DESCRIPTOR table. Given the OWNER and TABLE_NAME of the row, the ID_TABLE can be extracted in an insert database trigger and automatically filled before the insert takes place.

Specific sub-detectors may find they wish to add columns to these meta-data tables. We require only that the core columns as described here must exist in the schema, and that these additional columns not interfere with the original base columns. In particular, the CONDITIONS_SOURCE_DESCRIPTOR table would be a likely place to store additional sub-detector specific information.

The set of meta-data tables also describes the general type of "Value vs. Time" tables which will be common for DAQ, Trigger, Luminosity, and other monitoring, not just the PVSS/DCS sources of data. These tables will be implemented for those monitoring schema as well.

3. Mapping from SQL Query to Root *TTree*

Given a simple SQL query, one can map the resulting data onto a Root TTree object in a straightforward manner once the data-type mappings are defined. See the following table for a proposed map.

Oracle Data Type	Root Data Type	TTree tag	Leaf Type
NUMBER(1…11)	Int_t	/I	TLeafI
NUMBER(12…22)	Long64_t	/L	TLeafL
NUMBER(23…38) (?)	Double_t	/D	TLeafD
NUMBER(n.m) m>0	Double_t	/D	TLeafD
FLOAT	Float_t	/F	TLeafF
REAL	Double_t	/D	TLeafD
VARHCAR2(n)	Char_t	/C	TLeafC
DATE	TTimeStamp	(object)	TLeafElement
TIMESTAMP	TTimeStamp	(object)	TLeafElement
TIMESTAMP WITH TIME ZONE	TTimeStamp	(object)	TLeafElement

BLOB	?		
CLOB	?		

In general the database servlets available from the Web Based Monitoring home page at

http://cmsdaq.cern.ch/cmsmon/

automatically produce such a *root* file for the user to download to their local machine.